EXT: Simple Slideshow 2

Extension Key: simpleslideshow2
Language: en
Keywords: forEditors, forAdmins, forBeginners, simple, slideshow, flash, banner, rotater

Table of Contents

Introduction

What does it do?

Simple Slideshow 2 provides slideshow capability to Typo3 in the form of a plugin element that is inserted as content, just like any other type of content. The main benefit of Simple Slideshow 2 when compared with other slideshow extensions, is that files need only be uploaded to the selected folder on the server for them to be available to the slideshow; You may use FTP, the Filelist module, or any other method you can think of... If the image file is in the folder, it's available to be displayed by the slideshow automatically. You don't need to create individual records for each image.

Performs out-of-the-box as a non-clickable banner rotator. A banner rotator extension for Simple Slideshow 2 that allows defining links for particular images will be released in the near future.

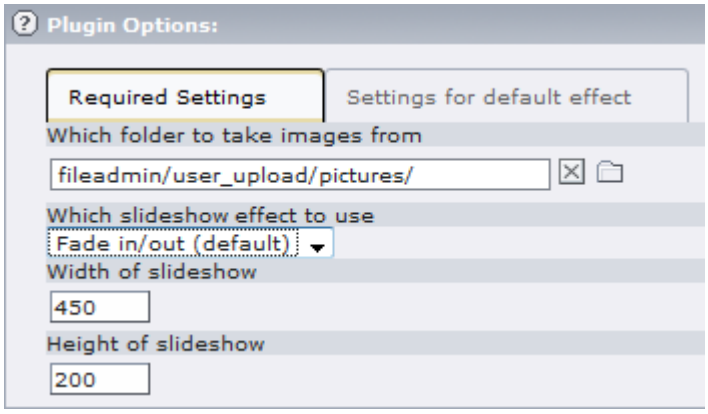
See the Youtube video screencast: http://www.youtube.com/watch?v=PnHFNm7v_9s

Typo3 Developers:

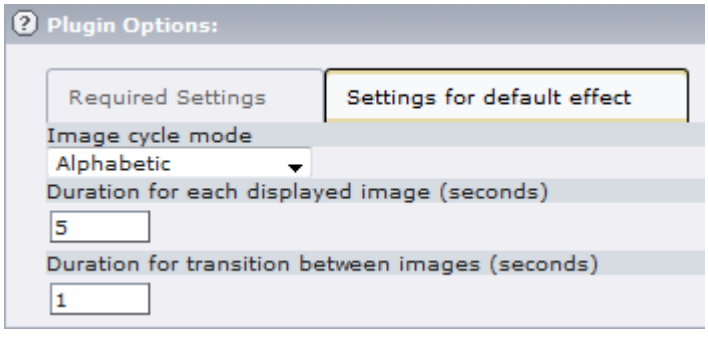
Simple Slideshow 2 is also extendable. This means it's possible to create multiple different methods of displaying images, whereby the same principle applies; If the image is in the folder, it's accessible to be displayed. To read more about this feature, see the Extension Development chapter.

Screenshots

I would have included some screenshots of how the slideshow actually looks in the front end, but it would just be a screenshot of an image, which isn't particularly useful, so here are some shots from the backend, instead.



The required settings tab

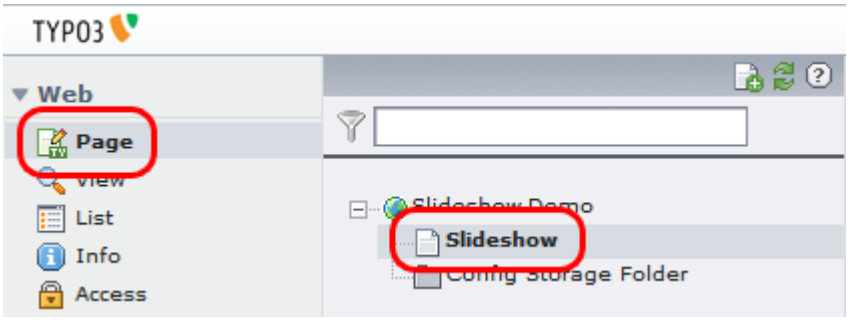


The settings tab for the default slideshow effect

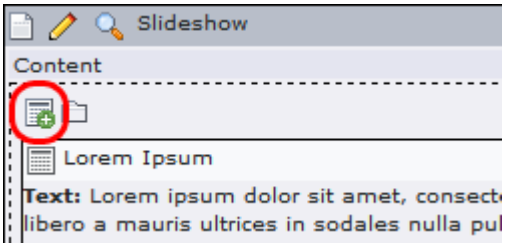
User's manual

This section of the manual is written with content editors in mind. It describes in easy steps how to add the slideshow to the page, configure it, and add and remove images for display. I will assume that you are using the Templavoila *Page* module.

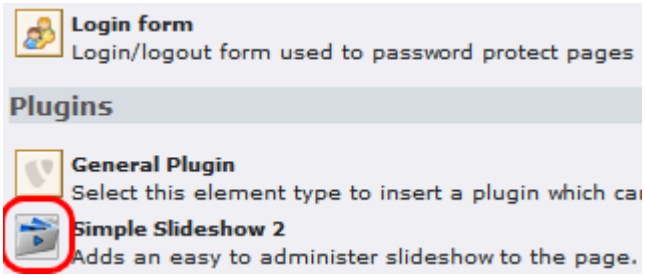
– To add the plugin to the page, click the Page module in the top left of the Typo3 backend, and then click the page in the pagetree you want to add the slideshow to.



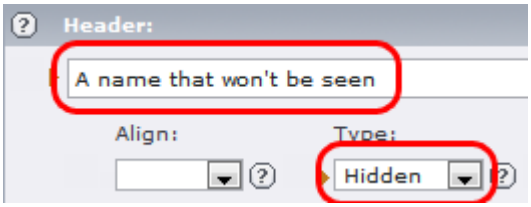
– In the main content area (the large area to the right of the pagetree), click the add content icon in the content field you would like the slideshow to appear in.



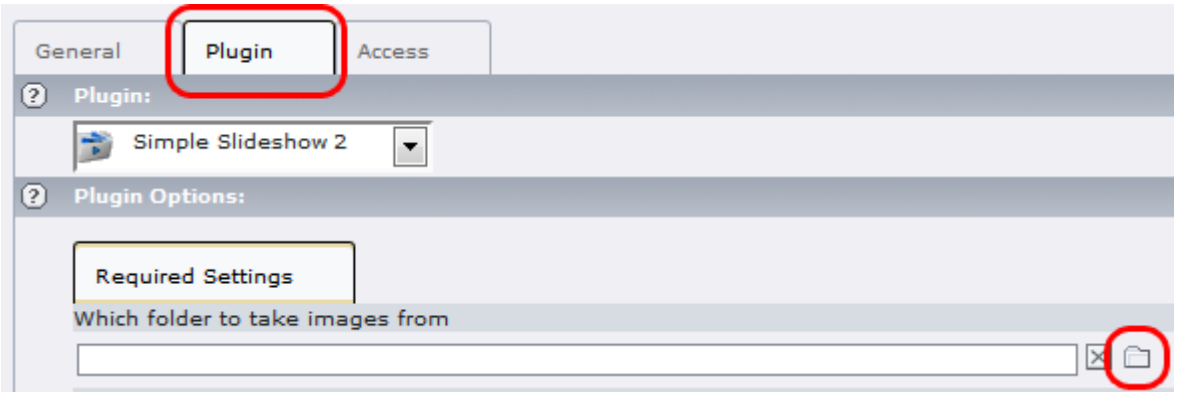
– Scroll down, find and click *Simple Slideshow 2.*



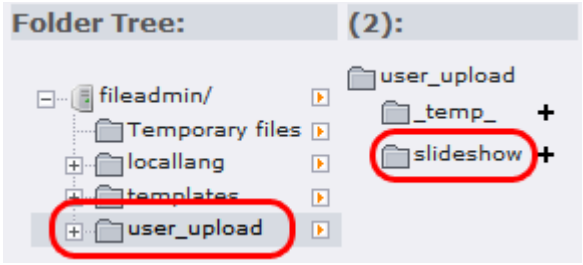
– Give the element a name, or select hidden as the display type if you do not wish the title to be displayed.



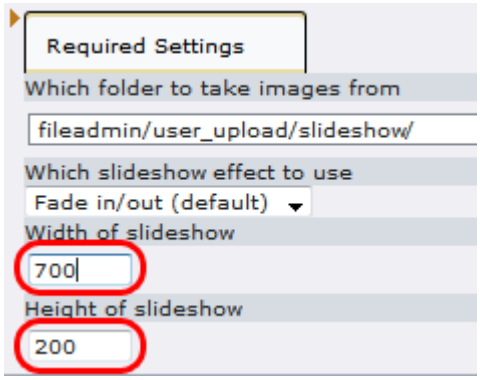
– Click the *Plugin* tab, and click the small folder icon at the top right of the *Required settings* tab.



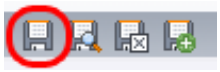
– Browse to the folder you would like to display images from, and click it. (This will very probably be a different folder on your installation)



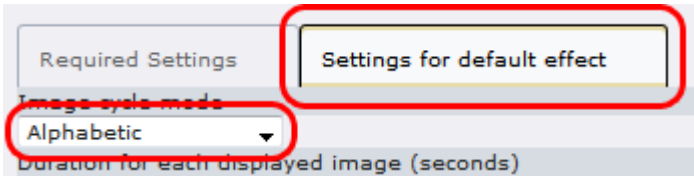
– Set the *width* and *height*, in pixels.



– Click the *Save* icon. This will load the settings tab for the default effect.



– Click the *Settings for default effect* tab, then click and select an *Image cycle mode*.



– Set the *duration for each displayed image*, and the *duration for transition between images.*

– Click the save and close button.

– View the page in the front end. (Click the page icon in the pagetree and click view from the popup menu)

– You should now see the slideshow displaying the images contained in the folder you selected.

Try uploading more images to the folder, or deleting images, and notice that it's entirely dynamic. You can upload images however you like... via FTP, through the Filelist module, via t3quixplorer, or even copy them directly in the shell on the server. If you don't want an image to be displayed in the slideshow, the only way of achieving this is to remove it from the selected folder. It's important to understand that every image in the selected folder is displayable by the slideshow! If you accidentally copy the wrong image into the folder, it will be displayed!

Administration

This section of the manual is aimed at Typo3 administrators. It assumes you have enough knowledge about the administration of Typo3 through the backend to be able to follow instructions without referring to screenshots.

To install Simple Slideshow 2:

- Import Simple Slideshow 2 from the Typo3 Extension Repository using Extension Manager. (extension key: simpleslideshow2)
- Once imported and installed, set the default allowed image types and which PID to use as the cache generator. (read more about this)
- Include the static TS template **Default Setup (simpleslideshow2)** to your site's root TS template.

Configuration

This section of the manual is aimed at advanced Typo3 administrators who understand Typoscript.

EM Settings

In the EM, there are two settings which have global influence over the extension. These are:

- **Allowed Image Extensions:** This setting is a pipe-separated list of allowed extensions. Only files with these extensions will be made available for display by the slideshow. By default allowed image types are: jpeg, jpg, png and gif.
- **Configuration PID:** If the cache file cannot be found, the Flash slideshow makes a call to a page on the site using a predetermined TypeNum to generate the cache file. This is the PID of the page to be used as the target for that call. By default, the root page has a PID of 1, but if you should have a need to specifically set a page (for example, if you only want to include the static template on the page the slideshow appears on), you may do that here.

Constants Reference

Constants available for **plugin.tx_simpleslideshow2_pi1**

Property:	Data type:	Description:	Default:
typeNum	integer	The page typeNum the cache generator is available on.	699

Setup Reference

Setup properties available for **plugin.tx_simpleslideshow2_pi1**

Property:	Data type:	Description:	Default:
cornersFile	string	A path to a png file. The cornersFile is a PNG file that is used as a mask for the four corners of the slideshow. The file must have a width and height that is divisible by 2. Transparent areas are masked in the flash. The file is sliced into 4 quarters by the flash, and each of the quarters is then positioned in its appropriate corner in the flash. If you look at the actual default corners file in Photoshop (or whatever image manipulation program you choose) you'll understand how easy it is.	EXT:simpleslideshow2/res/corners.png
maskFile	string	A path to a png file. If the mask file is defined, it overrides the cornersFile. The same principle applies: The transparent areas of the file are masked. By default, this setting is disabled.	EXT:simpleslideshow2/res/mask.png
defaultFolder	string	The path of the default folder the image files are served from	EXT:simpleslideshow2/res/
height	int	The default height of the slideshow (in pixels)	200
width	int	The default width of the slideshow (in pixels)	200
wrap	int	A standard wrap function, that surrounds the element with the specified text.	|

Setup properties available for **plugin.tx_simpleslideshow2_pi1.slideshowEffects.default**

Property:	Data type:	Description:	Default:
cycleMode	integer	Which mode to use to cycle through the images. 0 = alphabetic, 1 = shuffle, 2 = random, 3 = reverse.	0
delay	integer	The number of seconds each image should be displayed for.	5
transitionDelay	integer	The number of seconds each transition takes.	1

Please note: **plugin.tx_simpleslideshow2_pi1.slideshowEffects.default2** is not implemented in this version.

Tutorial

The best way of seeing how to use this extension is to watch the Youtube screencast here:

http://www.youtube.com/watch?v=PnHFNm7v_9s

Extension Development

One of the most useful features of Simple Slideshow 2, is that it's designed to be extendable. If you have an idea for a slideshow effect that you want to implement, then you can use the documentation in this chapter to realise your idea. You'll need to have reasonable proficiency in AS3, and a good understanding of creating Typo3 extensions.

In this chapter, I'll be using the extension *simpleslideshow2_static* as an example, so you'll need to make appropriate changes in the appropriate places. I'd suggest having its sources open in your IDE as you read this, for reference.

Overview

The basic idea of Simple Slideshow 2 is to create a cache file that outputs a JSON-ised configuration array. This is then read by the Flash. There are some default properties in the configuration array, and a custom sub-array, containing the configuration specific to the Simple Slideshow 2 extension that is being used to output the slideshow functionality. The default slideshow effect is also configured as a separate extension (though it is integrated), in this regard. You can see this by looking at the output from the cache file for a plugin configured to use the default effect. Here's an example of the JSON output from the site I took the screenshots from:

```
{"imageFiles":
["http:\/\/example.com\/fileadmin\/user_upload\/slideshow\/Chrysanthemum.jpg","http:\/\/example.com\/fil
eadmin\/user_upload\/slideshow\/Desert.jpg","http:\/\/example.com\/fileadmin\/user_upload\/slideshow\/Hy
drangeas.jpg","http:\/\/example.com\/fileadmin\/user_upload\/slideshow\/Jellyfish.jpg","http:\/\/example
.com\/fileadmin\/user_upload\/slideshow\/Koala.jpg","http:\/\/example.com\/fileadmin\/user_upload\/slide
show\/Lighthouse.jpg","http:\/\/example.com\/fileadmin\/user_upload\/slideshow\/Penguins.jpg","http:\/\/
example.com\/fileadmin\/user_upload\/slideshow\/Tulip.jpg"],"width":"700","height":"120","cornersFile":"
http:\/\/example.com\/typo3conf\/ext\/simpleslideshow2\/res\/corners.png","default":
{"cycleMode":"0","delay":"5","transitionDelay":"1"}}
```

As you can see, the last object is keyed as *default*. Therefore, in the AS3 class built to receive this configuration array (the default slideshow effect), it looks in the *default* object for its specific configuration.

The cache file itself checks to see if the folder containing the images has been modified each time it is called. If it has, then internally it calls the URL that generates a new cache file.

If the cache file doesn't exist, (each time you click the save button whilst in a Simple Slideshow 2 plugin element, it automatically deletes the cache file for that element) then the flash will use the name of the cache file to construct the URL it can call that generates a new cache file. The URL is available on any page on the *type* specified by the *plugin.tx_simpleslideshow2_pi1.typeNum* constant. The tt_content element UID must also be specified. A specific PID is also used, as set in the EM. So we end up with 3 parameters that build the cache filename, for example: 1_34_699.php. The Flash then reconstructs this into the URL that regenerates the cache:

http://example.com/index.php?id=1&elid=34&type=699

Thus, we have a robust and efficient caching mechanism, that enables the extension to dynamically use all the images in the specified folder, without the need to create database records or otherwise complicate matters.

Extensions of Simple Slideshow 2 hook into the JSON configuration process, and do whatever manipulation is necessary to get the required configuration options passed to the Flash. Therefore, each extension of Simple Slideshow 2 automatically benefits from this caching mechanism.

Required Files

- typo3conf/ext/simpleslideshow2_static/ext_localconf.php
- typo3conf/ext/simpleslideshow2_static/ext_tables.php
- typo3conf/ext/simpleslideshow2_static/locallang_db.xml
- typo3conf/ext/simpleslideshow2_static/slideshow_flexform.xml
- typo3conf/ext/simpleslideshow2_static/res/simpleslideshow2_static.swf

Other files, such as ext_icon.gif and ext_emconf.php will be made automatically by the kickstarter.

ext_localconf.php

```
<?php
if (!defined ('TYPO3_MODE')) {
      die ('Access denied.');
}

$GLOBALS['T3_VAR']['ext']['simpleslideshow2']['extensions']['simpleslideshow2_static'] = array(
      'jsonOutputHook' =>
'EXT:simpleslideshow2_static/class.tx_simpleslideshow2_static_jsonHook.php:tx_simpleslideshow2_static_js
onHook',
```

```
	'dropDownTitle' =>
'LLL:EXT:simpleslideshow2_static/locallang_db.xml:dropDownTitle.simpleslideshow2_static'
);
?>
```

Here, we set up a configuration array for the extension in the global *T3_VAR* array. Simple Slideshow 2 will scan its *extensions* subkey for registered extensions. Each registered extension must have two named elements: *jsonOutputHook* and *dropDownTitle*.

jsonOutputHook registers a hook for processing the json array before it's passed to the Flash. Though this isn't 100% necessary, I can't imagine that you wouldn't need to maniplate the json in some way, so in nearly all cases, it will be necessary, and you'll need to create the file and class.

dropDownTitle is a string (that supports LLL:) that represents the readable name of the extension in the Simple Slideshow 2's plugin options in the "Which slideshow effect to use" dropdown.

ext_tables.php

```
<?php
if (!defined ('TYPO3_MODE')) {
	die ('Access denied.');
}
t3lib_extMgm::addStaticFile($_EXTKEY,'static/default_setup/', 'Default Setup');
?>
```

Here, we simply add a static template to the system, as there are some settings used in TS by the extension.

locallang_db.xml

This is just a regular locallang file contianing key/value definitions.

slideshow_flexform.xml

The absolutely crucial thing with this flexform is that the name of the only sheet is exactly the same as the extension key. So, in this example, it is:

```
<sheets>
	<simpleslideshow2_static>
	…
```

Other sheets will be ignored.

res/simpleslideshow2_static.swf

This is a compiled Flash class that deserves its own chapter. See the AS3 Class chapter for more information.

Included Files

– typo3conf/ext/simpleslideshow2_static/class.tx_simpleslideshow2_static_jsonHook.php

– typo3conf/ext/simpleslideshow2_static/static/default_setup/setup.txt

As we've seen in ext_localconf.php, there is an included file for the json hook. There's also a reference to an included static template.

class.tx_simpleslideshow2_static_jsonHook.php

```
function jsonHook(&$arrJson, $effectKey, $row, $obj) {

	//Get the TS config
	$conf = $GLOBALS['TSFE']->tmpl->setup['plugin.']['tx_simpleslideshow2_static.'];

	$arrTmp = array();

	//Get specified values in the flexform, or set default values
	$arrTmp['fadeInDelay'] = ($tmp = $obj->pi_getFFvalue($row['pi_flexform'], 'fadeInDelay',
$effectKey)) ? $tmp : $conf['fadeInDelay'];

	//arrJson is passed by reference, so we add the configuration variables we just set to a sub-
object, keyed using the passed effectKey variable.

	//In the AS3 code, the code will need to parse this object.
	$arrJson[$effectKey] = $arrTmp;
}
```

The hook is passed 4 variables:

– &$arrJson: Passed by reference. Operate on it directly to modify its contents.

– $effectKey: The same as the extension key.

– $row: The row from tt_content for this plugin element.

– $obj: The plugin object.

In this function, all we're doing is checking for an existing value, and setting it from the values set in the static TS template if it's missing, and making that change directly on the json array. Note how we're placing the values specific to this extension in their own sub-array in the json array. The AS3 is then coded to examine that array for its relevant settings. This prevents variable pollution.

setup.txt

```
plugin.tx_simpleslideshow2_static {
      fadeInDelay = 99
}
```

Here there's a single property that defines the time taken for the fade-in to occur, which is used by the function listed above.

AS3 Class

The AS3 must follow certain rules. The FLA file must include a spearate .AS file containing a class named SlideShow, and the class must be declared in the FLA. Here's a listing of the AS3 for the *simpleslideshow2_static* extension.

Code Listing

simpleslideshow2_static.fla

```
var blah:SlideShow;
```

SlideShow.as

```
package {

	import com.greensock.TweenMax;
	import flash.events.*;
	import flash.display.*;
	import flash.utils.*;
	import flash.net.*;

	public class SlideShow extends Sprite {

		private var config:Object;
		private var stg:Object;

		private var slideA:Sprite = new Sprite();
		private var slideB:Sprite = new Sprite();
		private var imgLoaderA = new Loader();
		private var imgLoaderB = new Loader();
		private var imgListPointer:int = -1;

		private var t:Timer;

		public function SlideShow(config:Object) {

			this.config = config;
			this.stg = config['stage'];

			trace("SlideShow");

			if(this.config['imageFiles'] == undefined) {
				this.stg.notifyError("No list of images to display");
				return;
			}

			if(getQualifiedClassName(this.config.imageFiles) != "Array") {
				this.stg.notifyError("No list of images to display");
				return;
			}

			if(this.config['simpleslideshow2_static'] == undefined) {
				this.config['simpleslideshow2_static'] = new Object();
				this.config.simpleslideshow2_static['fadeInDelay'] = 1;
			}

			if(getQualifiedClassName(this.config.simpleslideshow2_static) != "Object") {
				this.config['simpleslideshow2_static'] = new Object();
				this.config.simpleslideshow2_static['fadeInDelay'] = 1;
			}

			if(this.config.simpleslideshow2_static['fadeInDelay'] == undefined) {
				this.config.simpleslideshow2_static['fadeInDelay'] = 1;
			}

			addChild(this.slideA);
			this.slideA.addChild(this.imgLoaderA);
			this.slideA.alpha = 0;
			this.imgLoaderA.contentLoaderInfo.addEventListener(Event.COMPLETE, firstImgLoaded);
			this.imgLoaderA.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR,
loaderError);
			this.imgListPointer = this.getRandomIndex();
			this.imgLoaderA.load(new URLRequest(this.config.imageFiles[this.imgListPointer]));
		}

		private function firstImgLoaded(evt:Event):void {

			TweenMax.to(slideA, this.config.simpleslideshow2_static.fadeInDelay, {alpha: 1});
		}
```

```
            private function getRandomIndex() {

                  var tmp:int = this.imgListPointer;

                  if(this.config.imageFiles.length == 1) {
                        return 0;
                  }

                  while(tmp >= this.config.imageFiles.length || tmp == this.imgListPointer) {
                        tmp = Math.round(Math.random() * (this.config.imageFiles.length - 1));
                  }
                  return tmp;

            }

            function loaderError(evt:IOErrorEvent):void {

                  var tgt:Object = evt.target;
                  var file = IOErrorEvent(evt).toString().match(/URL: (.+?)"/);
                  trace("Couldn't load file: " + file[1]);

                  for(var i:* in this.config.imageFiles) {

                        if(this.config.imageFiles[i] == file[1]) {

                           trace("Removed " + this.config.imageFiles[i] + " from list of images to
display");
                           this.config.imageFiles.splice(i, 1);
                           this.imgListPointer = this.getRandomIndex();
                        }
                  }
                  tgt.loader.load(new URLRequest(this.config.imageFiles[this.imgListPointer]));
            }

      }
}
```

Code Analysis

In the .fla file, we define the class. If you don't do this, the class won't be accessible to the main flash file the compiled extension SWF is loaded into.

The resulting SWF file **must** be named the same as the extension key. Therefore, in this instance, the filename is simpleslideshow2_static.swf

In the SlideShow class, there are a few important things to notice. The class must extend sprite (or another DisplayObject object) as it's added to the stage with an addChild statement. Therefore, any displayable elements in the class must be added to the object using addChild, also.

The config object has a "stage" element, that is a reference to the stage object. There is a function on the stage named notifyError which accepts a string. As seen in the code, the first two error checks call this function to inform the user there's a problem.

Other than that, you can do what you want with the code. For development, I used the following stub code in the .fla file:

```
import com.adobe.serialization.json.JSON;

var dummyJSON:String = '{"imageFiles":["http:\/\/example.com\/fileadmin\/user_upload\/Billeder\/Forsiden
Blah\/Jellyfish.jpg","http:\/\/example.com\/fileadmin\/user_upload\/Billeder\/Forsiden
Blah\/Koala.jpg","http:\/\/example.com\/fileadmin\/user_upload\/Billeder\/Forsiden
Blah\/Lighthouse.jpg","http:\/\/example.com\/fileadmin\/user_upload\/Billeder\/Forsiden
Blah\/Penguins.jpg","http:\/\/example.com\/fileadmin\/user_upload\/Billeder\/Forsiden
Blah\/Tulips.jpg"],"width":"464","height":"100","cornersFile":"http:\/\/example.com\/typo3conf\/ext\/sim
pleslideshow2\/res\/corners.png","customSlideShow":"http:\/\/example.com\/typo3conf\/ext\/simpleslidesho
w2_static\/res\/simpleslideshow2_static.swf","simpleslideshow2_static":{"fadeInDelay":"5"}}';

var config:Object = JSON.decode(dummyJSON);
config['stage'] = this;

var whatever:SlideShow = new SlideShow(config);

addChild(whatever);

function notifyError(error:String):void {
      var warning:TextField  =  new TextField();
      warning.autoSize = TextFieldAutoSize.LEFT;
      addChild(warning);
      warning.text = "Error: "+error;
}
```

Known problems

None. So far.

To-Do list

- Perhaps convert it from a plugin to a new Ctype, which makes more logical sense, it's just more challenging to implement, what with the dynamic flexforms, and all...
- Add a couple more default slideshow effects, rather than just the one that's implemented in this version.

ChangeLog

Version	Changes:
0.0.1	Initial upload to the TER.